



Docket No.: 50023-135

# 67

PATENT

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Application of

Hideaki MIYAKE, et al.

Serial No.: 09/812,894

Group Art Unit: 2133

Filed: March 21, 2001

Examiner:

For: DEBUGGING SUPPORTING APPARATUS, DEBUGGING SUPPORTING  
METHOD AND RECORDING MEDIUM READABLE BY COMPUTER WITH  
ITS PROGRAMS RECORDED THEREON

**TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENTS**

Honorable Commissioner for Patents and Trademarks  
Washington, D. C. 20231

Sir:

At the time the above application was filed, priority was claimed based on the  
following applications:

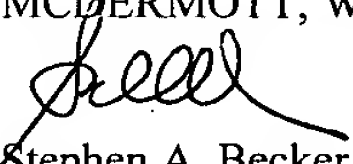
Japanese Patent Application No. 2000-094064, Filed March 30, 2000; and

Japanese Patent Application No. 2000-375641, December 11, 2000

A copy of each priority application listed above is enclosed.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

  
Stephen A. Becker  
Registration No. 26,527

600 13<sup>th</sup> Street, N.W.  
Washington, DC 20005-3096  
(202) 756-8000 SAB:ykg  
**Date: June 4, 2001**  
Facsimile: (202) 756-8087



日 本 国 特 許 庁  
PATENT OFFICE  
JAPANESE GOVERNMENT

50023-135  
June 4, 2001  
MIYAKE, ET AL.  
McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 3月30日

出 願 番 号

Application Number:

特願2000-094064

出 願 人

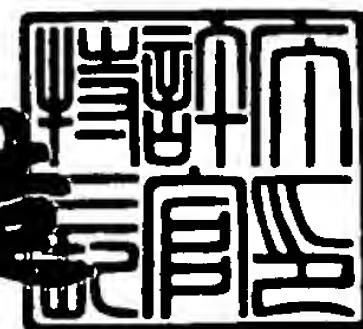
Applicant(s):

松下電器産業株式会社

2000年12月15日

特 許 庁 長 官  
Commissioner,  
Patent Office

及 川 耕 造



出証番号 出証特2000-3104154

【書類名】 特許願

【整理番号】 2022510576

【提出日】 平成12年 3月30日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 11/28

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 三宅 秀明

【発明者】

【住所又は居所】 東広島市鏡山3丁目10番18号 株式会社松下電器情報システム広島研究所内

【氏名】 枯木 正吉

【発明者】

【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式会社内

【氏名】 鶴本 克己

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100083172

【弁理士】

【氏名又は名称】 福井 豊明

【手数料の表示】

【予納台帳番号】 009483

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

特 2 0 0 0 - 0 9 4 0 6 4

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9713946

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 デバッグ支援システムおよびデバッグ支援方法

【特許請求の範囲】

【請求項 1】 OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、OSシミュレータが管理するOS管理情報をOSデバッガから操作できるようにしたデバッグ支援システムにおいて、

上記OSシミュレータの要請に応じてOS管理情報が書き込まれるとともに、上記OSデバッガの要請に応じてOS管理情報が読み出される共有ファイルを備えたことを特徴とするデバッグ支援システム。

【請求項 2】 上記共有ファイルにOS管理情報を書き込むOS管理情報書き込み手段をアプリケーション側に備えるとともに、

上記共有ファイルからOS管理情報を読み出すOS管理情報読み出し手段をOSデバッガ側に備えた請求項 1 に記載のデバッグ支援システム。

【請求項 3】 上記OS管理情報読み出し手段が、任意のタイミングで共有ファイルからOS管理情報を読み出す請求項 2 に記載のデバッグ支援システム。

【請求項 4】 上記OS管理情報読み出し手段が、所定の周期で共有ファイルからOS管理情報を読み出す請求項 2 に記載のデバッグ支援システム。

【請求項 5】 OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、OSシミュレータが管理するOS管理情報をOSデバッガから操作できるようにしたデバッグ支援システムにおいて、

上記OSデバッガの要請に応じてOS管理情報を変更するためのデバッグ命令が書き込まれるとともに、上記アプリケーションの要請に応じてデバッグ命令が読み出される共有ファイルを備えたことを特徴とするデバッグ支援システム。

【請求項 6】 上記共有ファイルにデバッグ命令を書き込むデバッグ命令書き込み手段をOSデバッガ側に備えるとともに、

上記共有ファイルからデバッグ命令を読み出すデバッグ命令読み出し手段をアプリケーション側に備えた請求項 5 に記載のデバッグ支援システム。

【請求項 7】 OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援システムにおいて、

上記OSシミュレータが管理するOS管理情報を変更するためのデバッグ命令が書き込まれた共有ファイルと、該共有ファイルからデバッグ命令を読み出すデバッグ命令読み出し手段とを備えたことを特徴とするデバッグ支援システム。

【請求項8】 上記デバッグ命令読み出し手段が、任意のタイミングで共有ファイルからデバッグ命令を読み出す請求項6又は7に記載のデバッグ支援システム。

【請求項9】 上記デバッグ命令読み出し手段が、所定の周期で共有ファイルからデバッグ命令を読み出す請求項6又は7に記載のデバッグ支援システム。

【請求項10】 OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、OSシミュレータが管理するOS管理情報をOSデバッガから操作できるようにしたデバッグ支援方法において、

OSシミュレータとOSデバッガとの間に共有ファイルを介在させることによって上記操作を可能としたことを特徴とするデバッグ支援方法。

【請求項11】 OSシミュレータが管理するOS管理情報をOSデバッガから操作しながら、OSシミュレータによるシミュレーション環境で動作するアプリケーションを作成する方法において、

OSシミュレータとOSデバッガとの間に共有ファイルを介在させることによって上記操作を可能としたことを特徴とするアプリケーションの作成方法。

【請求項12】 OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、

OSシミュレータとOSデバッガとの間に共有ファイルを介在させることによって、OSシミュレータが管理するOS管理情報をOSデバッガから操作できるようにするためのプログラムをコンピュータ読み取り可能な形態で記録したことを特徴とする記録媒体。

【発明の詳細な説明】

【発明の属する技術分野】

本発明は、デバッグ支援システムに関し、特に、OSシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援システムに関するものである。

## 【従来の技術】

近年の携帯電話など（以下「機器」という）には、図4（a）に示すように、様々な機能を実現するためのアプリケーション10が組み込まれており、このアプリケーション10は、通常、機器専用のオペレーティングシステム（以下「専用OS」という）20a上で動作するようになっている。従って、このような組み込み型のアプリケーション10の開発は、機器のハードウェアや専用OS20aの開発と並行して行えるよう、図4（b）に示すように、汎用コンピュータ上のマルチプロセス環境（すなわち汎用OS50上）に構築した組み込み先のシミュレーション環境において行う。

上記シミュレーション環境は、専用OS20aの動作（例えば、タスク管理動作や、イベントフラグなどのリソース管理動作）をシミュレートしたOSシミュレータ20をライブラリという形式でアプリケーション10にリンクさせることによって構築することができる。

ここで、アプリケーション10のデバッグは、汎用のアプリケーション開発ツール（図示せず）に含まれるアプリケーションデバッガ30を用いて行うようになっている。このデバッグとは、プログラム中のバグを取り除く作業をいい、デバッガとは、デバッグを効率よく行えるようにしたユーティリティプログラムをいう。

しかしながら、上記シミュレーション環境で開発したアプリケーション10をデバッグする場合、アプリケーション10の実行中におけるタスクの状態や、このタスクの状態を変化させるイベントの有無を1または0のビットパターンで表したイベントフラグなど、図示しないメモリにおいてOSシミュレータ20が管理する各種情報（以下「OS管理情報」という）を参照あるいは変更したいときがある。すなわち、アプリケーションデバッガ30は、上記したように汎用のアプリケーション開発ツールであるため、OS管理情報を参照あるいは変更（以下、両者を一括して「操作」という）するという特別な機能は備えていない。

そこで、最近では、アプリケーションデバッガ30だけでなく、OS管理情報を操作するためのユーティリティプログラム（以下「OSデバッガ40」という）を備えたデバッグ支援システムが登場している。

このOSデバッガ40は、図3に示すように、OSシミュレータ20とは別のプロセス空間に存在するため、OS管理情報に直接アクセスすることはできない。従って、アプリケーション10側とOSデバッガ40側の両方に通信手段C<sub>1</sub>・C<sub>2</sub>を備え、これら通信手段C<sub>1</sub>・C<sub>2</sub>がプロセス間通信を行うことによって、OSデバッガ40からOS管理情報を操作できるようにしている。

【発明が解決しようとする課題】

ところで、ブレークポイントを設定するなどしてアプリケーション10を停止させ、この時点のOS管理情報を参照できれば、バグの位置を見当付けるのが容易になる。

しかしながら、アプリケーション10が停止した場合は、このアプリケーション10とリンクしているOSシミュレータ20も停止する。すなわち、上記従来のデバッグ支援システムのようにプロセス間通信を採用した構成によれば、アプリケーション10が停止すると、OSシミュレータ20とOSデバッガ40との間で通信が行えなくなり、この時点のOS管理情報を参照できないという問題があった。

この問題は、ブレークポイントを設定してアプリケーション10を意識的に停止させた場合だけでなく、アプリケーション10がハングアップした場合も同様に発生する（ハングアップしたの時点のOS管理情報を参照できなければ、その原因を特定するのは難しい）。

本発明は上記従来の事情に基づいて提案されたものであって、OSシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のOS管理情報を参照できるようにしたデバッグ支援システムを提供することを目的とするものである。

【課題を解決するための手段】

本発明は上記目的を達成するために以下の手段を採用している。すなわち、本発明は、図1に示すように、OSシミュレータ20によるシミュレーション環境で開発したアプリケーション10をデバッグする際、OSシミュレータ20が管理するOS管理情報をOSデバッガ40から操作できるようにしたデバッグ支援システムを前提としている。

ここで、共有ファイルFには、上記OSシミュレータ20の要請に応じてOS管理情報が書き込まれ、このOS管理情報は、上記OSデバッガ40の要請に応じて読み出される。このようにすれば、アプリケーション10が停止した場合でも、この時点のOS管理情報を上記共有ファイルFから参照することができる。

また、図2に示すように、OS管理情報を変更するためのデバッグ命令を共有ファイルFに書き込むとともに、この共有ファイルFからデバッグ命令を読み出すようにすれば、OS管理情報を変更できる。

### 【発明の実施の形態】

以下に本発明の実施の形態を図面に従って詳細に説明する。

#### (第1の実施の形態)

図1は、本発明を適用したデバッグ支援システムの概略機能ブロック図であり、以下その構成を上記従来と異なる点のみ説明する。

まず、図示しないアプリケーションデバッガ30を用いてアプリケーション10を実行すると、例えば、タスクAがシステムコール「set \_\_flg」を発行し、このシステムコール「set \_\_flg」を受け付けたOSシミュレータ20のシステムコール処理手段21は、イベントフラグの値を0から1に変更するようOS管理情報変更手段22に指示する。

これによって、OS管理情報変更手段22は、図示しないメモリに格納されているイベントフラグの値を0から1に変更した後、この変更内容をアプリケーション10に含まれるOS管理情報書き込み手段12に通知し、この通知を受けたOS管理情報書き込み手段12は、イベントフラグの値が0から1に変更された旨を共有ファイルFに書き込む。なお、共有ファイルFとは、複数のプロセスによって同時に共有されるファイルをいう。

ここで、アプリケーション10が停止し、この時点のOS管理情報を参照したい場合は、OSデバッガ40のユーザインターフェイス41を介して「イベントフラグの値参照」を指示する。

これによって、メイン処理手段42は、イベントフラグの値を読み出すようOS管理情報読み出し手段43に指示し、この指示を受けたOS管理情報読み出し手段43は、共有ファイルFからイベントフラグの値を読み出して表示処理手段

4 4 に渡し、更に、表示処理手段 4 4 は、このイベントフラグの値をユーザが視覚的に理解できるような形式に変換して表示する。

以上のように、本発明を適用したデバッグ支援システムによれば、イベントフラグの値を共有ファイル F に書き込むようにしているため、アプリケーション 1 0 が停止した場合でも、この時点のイベントフラグの値を参照することができる。

なお、イベントフラグの値を共有ファイル F から読み出すタイミングは、「イベントフラグの値参照」が指示されたときに限定されるものではない。すなわち、OS 管理情報読み出し手段 4 3 に対して周期的に読み出しを指示する手段を備えた構成としてもよい。このようにすれば、OS 管理情報の時間的遷移を知ることができるため、予期しないアプリケーション 1 0 の状態を容易に発見することができる。

また、OS 管理情報の種別毎に共有ファイル F の使用領域を決めておけば、イベントフラグの値だけでなく、他の OS 管理情報（タスクの状態・セマフォの状態・メールボックスの状態など）も同様の手順で参照できることはいうまでもない。

#### （第 2 の実施の形態）

ところで、上記のように参照したイベントフラグの値が 1 である場合は、バグがないものとしてアプリケーション 1 0 の実行を再開すればよいが、0 である場合は何らかのバグがあり、このバグを取り除く必要がある。

しかしながら、上記デバッグが完了するまで残りの動作検証が行えないのでは効率が悪い。すなわち、まずはアプリケーション 1 0 の全部について動作検証をし、その後、一括してデバッグするのが好ましい。

そこで、本実施の形態では、OS 管理情報が不正な値となった場合は、この OS 管理情報を OS デバッガ 4 0 から変更できるようにしており、以下その構成を図 2 に従って説明する。

例えば、イベントフラグの値を 0 から 1 に変更したい場合は、OS デバッガ 4 0 のユーザインターフェイス 4 1 を介して「デバッグ命令」を指示し、更に、このデバッグ命令のうちの「イベントフラグの値変更（変更後の値：1）」を指示

する。これによって、メイン処理手段 4 2 は、イベントフラグの値を 0 から 1 に変更するようデバッグ命令書き込み手段 4 5 に指示し、この指示を受けたデバッグ命令書き込み手段 4 5 は、イベントフラグの値を 0 から 1 に変更するシステムコール「set \_\_flg」を、それに対応するデバッグ命令機能コード(-48)に変換して共有ファイル F に書き込む。

一方、アプリケーション 1 0 に含まれるデバッグ命令読み出し手段 1 5 は、タイマ 1 6 によって共有ファイル F を周期的に参照するようになっており、上記のようにデバッグ命令機能コード(-48)が共有ファイル F に書き込まれると、このデバッグ命令機能コード(-48)を読み出してシステムコール情報生成手段 1 4 に渡す。

ここで、システムコール情報生成手段 1 4 は、上記デバッグ命令機能コード(-48)に対応するシステムコール情報「set \_\_flg」を生成してデバッグタスク起動手段 1 3 に渡し、更に、デバッグタスク起動手段 1 3 は、このシステムコール情報「set \_\_flg」を指定してデバッグタスク 1 7 を起動する。なお、システムコール情報「set \_\_flg」を指定する方法は、デバッグタスク起動のパラメータや共有ファイル F を用いる方法など様々あり、特に限定されるものではない。

その後、上記のように起動されたデバッグタスク 1 7 が OS シミュレータ 2 0 に対してシステムコール「set \_\_flg」を発行し、これによって、メモリに格納されているイベントフラグの値が 0 から 1 に変更されると、OS シミュレータ 2 0 のタスク管理手段（図示せず）が、イベントフラグ待ちの状態にあるタスク B を実行状態に遷移させる。

以上のように、本実施の形態によれば、何らかのバグが存在することによって OS 管理情報が不正な値となった場合でも、この OS 管理情報を OS デバッガ 4 0 から変更することができる。また、このような変更機能を利用すれば、アプリケーション 1 0 のプログラムを修正することなく、割り込みが発生した状況をつくることも可能である。

なお、ここでは、OS デバッガ 4 0 からデバッグ命令を指示することとしているが、本発明はこれに限定されるものではない。すなわち、予めデバッグ命令を共有ファイル F に格納しておき、このデバッグ命令をデバッグ命令読み出し手段

15 が周期的に（或いは、任意のタイミングで）読み出すようにしてもよい。このようにすれば、OS デバッガ 40 を用いることなく OS 管理情報を変更できる。

また、デバッグ命令を実行する手段はデバッグタスク 17 に限定されるものではない。すなわち、ここでは、デバッグタスク 17 がシステムコールを発行することによってデバッグ命令が実行される動作を例示しているが、システムコールには、タスクからのみ発行可能・割り込みハンドラからのみ発行可能・この両方から発行可能という種別があるため、この種別に応じた実行手段を選択するようにしている。

更に、上記の説明では、デバッグタスク 17 の状態遷移について特に言及していないが、システムコールを発行し終えたデバッグタスク 17 は、消滅しても、常駐しても（すなわち、次のデバッグ命令が共有ファイル F に書き込まれるまで待ち状態となっても）、いずれであってもかまわない。

#### 【発明の効果】

以上のように、本発明によれば、効果的なデバッグを行うことができる。

すなわち、第 1 の実施の形態によれば、OS 管理情報を共有ファイルに書き込むようにしているため、アプリケーションが停止した場合でも、この時点の OS 管理情報を OS デバッガから参照することができる。従って、バグの位置を見当付けるためにブレークポイントを設定することや、ハングアップの原因を容易に特定することが可能である。

また、周期的に OS 管理情報を参照するようにしておけば、予期しないアプリケーションの状態を容易に発見することができる。

一方、第 2 の実施の形態によれば、OS 管理情報を OS デバッガから変更できるため、OS 管理情報が不正な値となった場合でも、アプリケーションの動作検証を継続することや、アプリケーションのプログラムを修正することなく割り込みが発生した状況をつくることが可能である。

また、予めデバッグ命令を共有ファイルに格納しておけば、OS デバッガを用いることなく OS 管理情報を変更することができる。

#### 【図面の簡単な説明】

【図 1】

第 1 の実施の形態におけるデバッグ支援システムの概略機能ブロック図。

【図 2】

第 2 の実施の形態におけるデバッグ支援システムの概略機能ブロック図。

【図 3】

従来におけるデバッグ支援システムの概略機能ブロック図。

【図 4】

組み込み型アプリケーション 1 0 の開発環境の説明図。

【符号の説明】

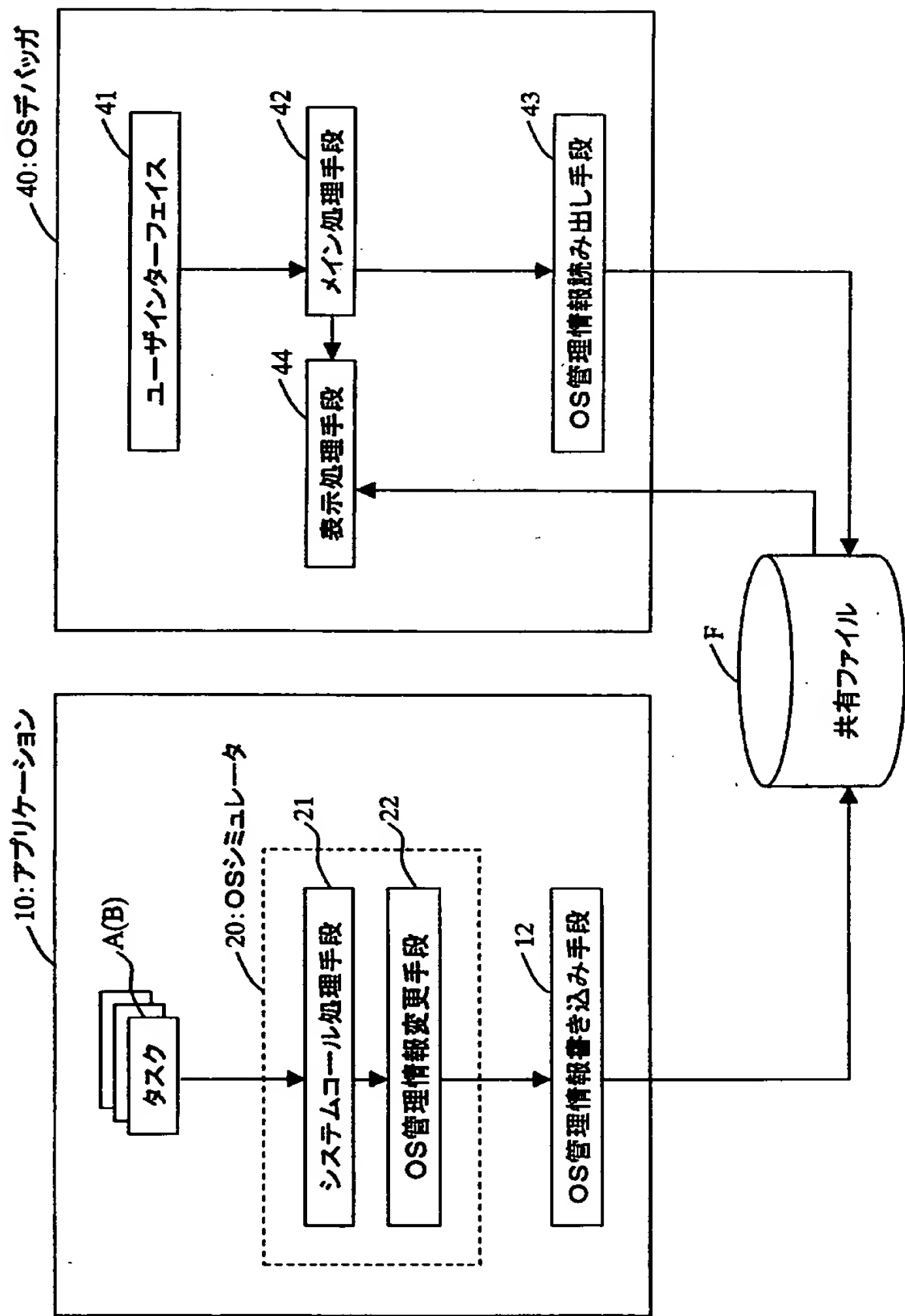
- 1 0      アプリケーション 1 0
- 1 2      OS 管理情報書き込み手段
- 1 3      デバッグタスク起動手段
- 1 4      システムコール情報生成手段
- 1 5      デバッグ命令読み出し手段
- 1 7      デバッグタスク
- 2 0      OS シミュレータ 2 0
- 2 1      システムコール処理手段 2 1
- 2 2      OS 管理情報変更手段
- 3 0      アプリケーション 1 0 デバッガ
- 4 0      OS デバッガ 4 0
- 4 1      ユーザインターフェイス
- 4 2      メイン処理手段
- 4 3      OS 管理情報読み出し手段
- 4 4      表示処理手段
- 4 5      デバッグ命令書き込み手段
- A, B    タスク

【書類名】

図面

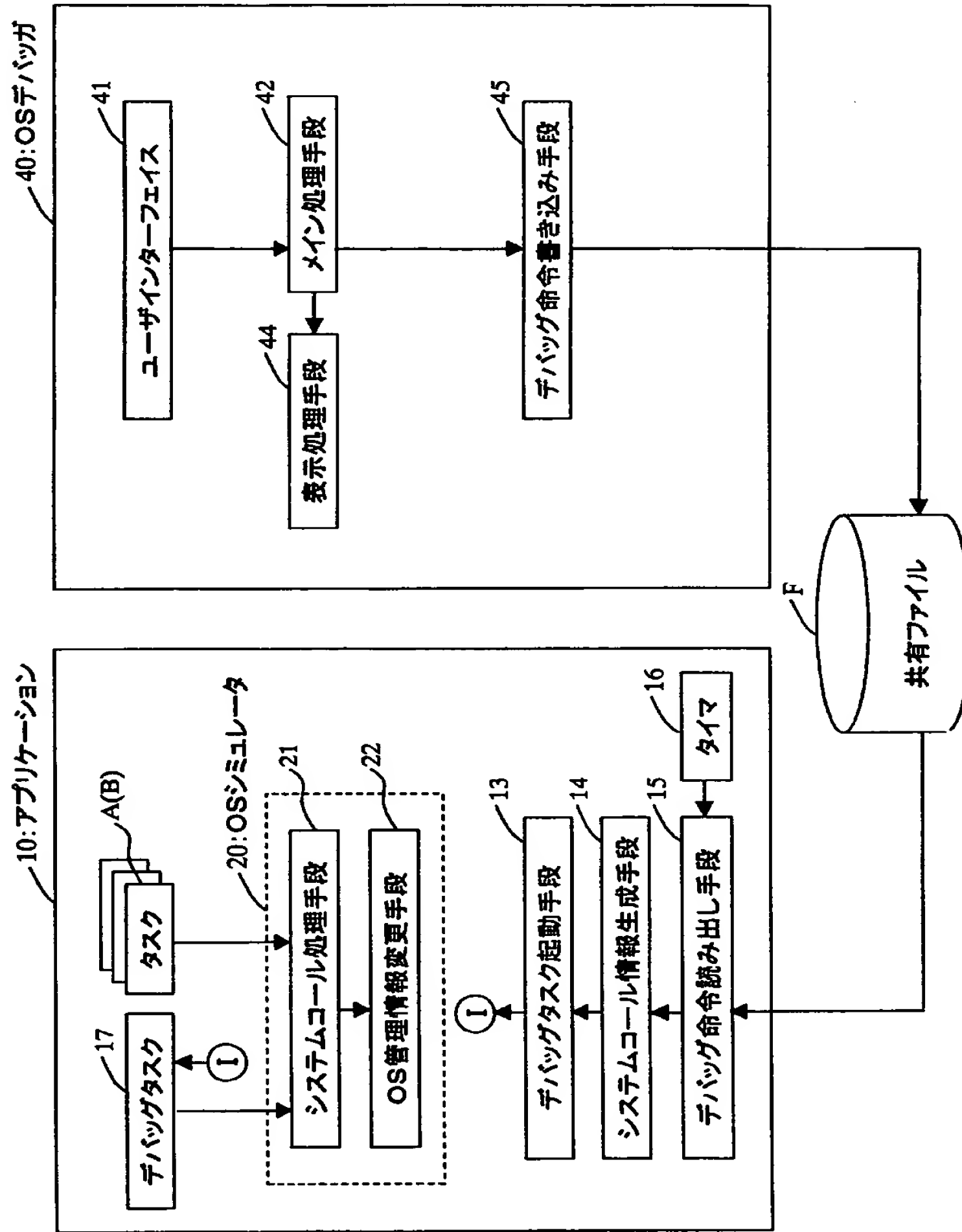
【図 1】

第1の実施の形態におけるデバッグ支援システムの概略機能ブロック図



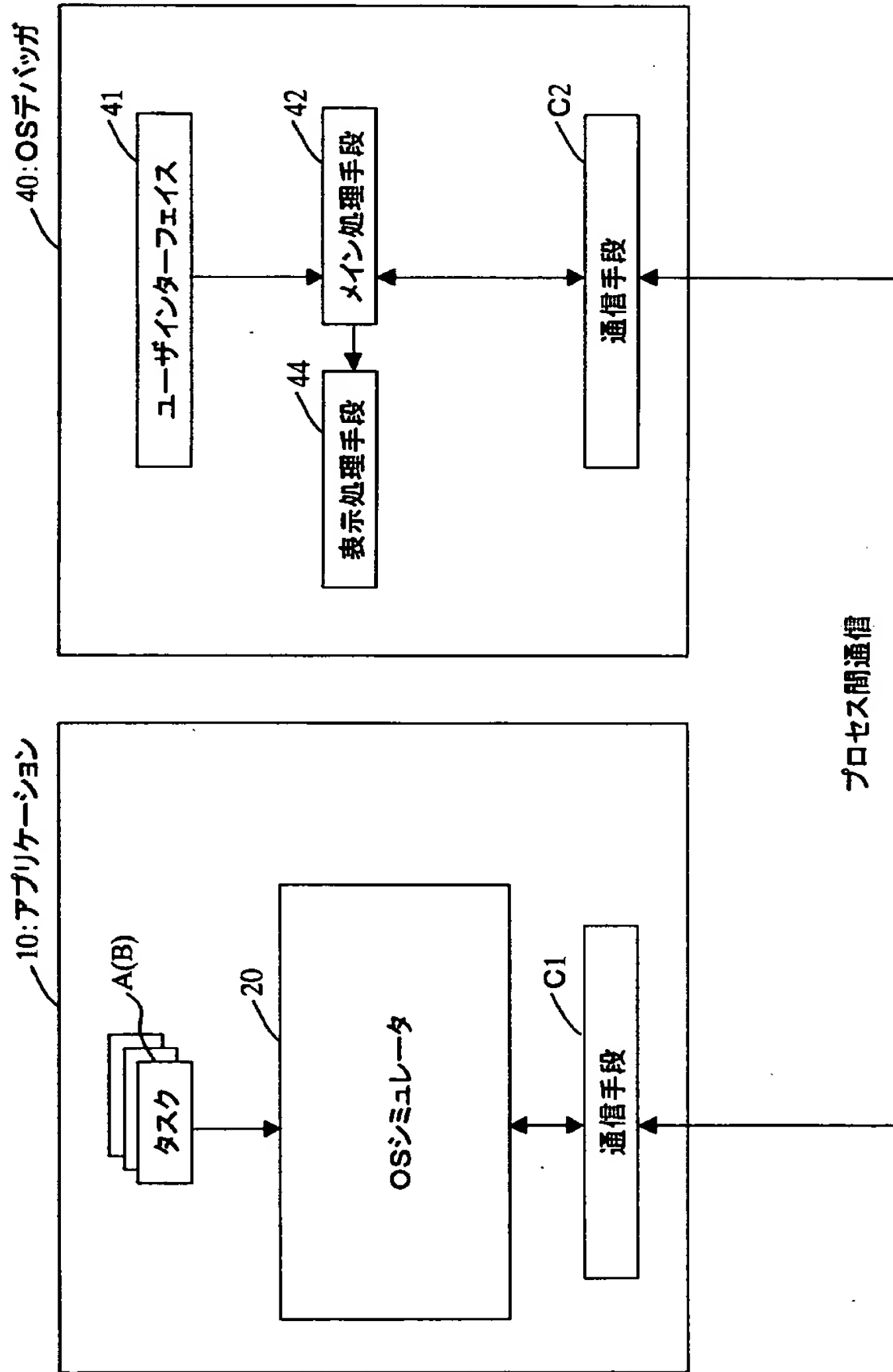
【図 2】

第2の実施の形態におけるデバッグ支援システムの概略機能ブロック図



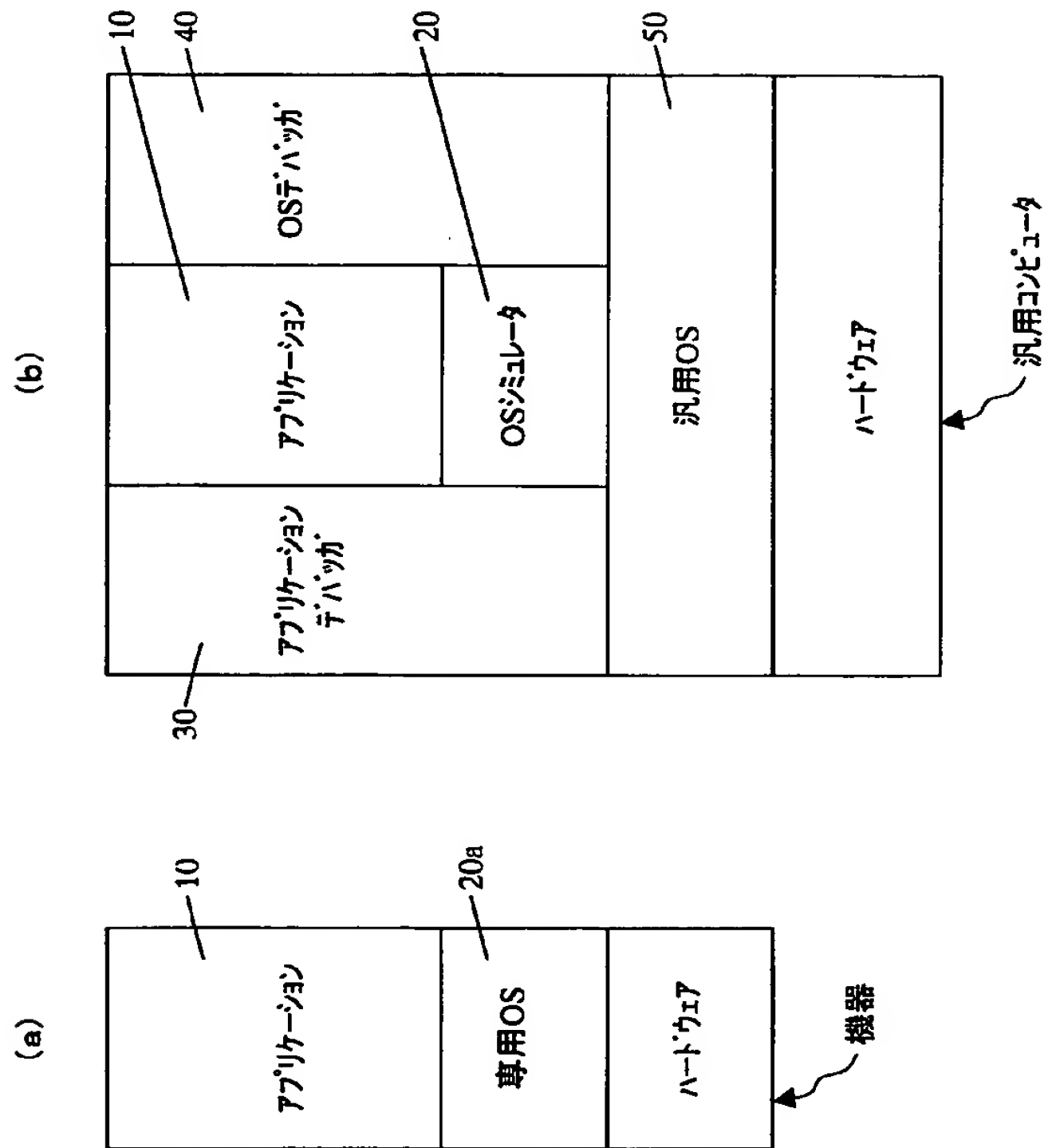
【図 3】

従来におけるデバッグ支援システムの概略機能ブロック図



【図 4】

組み込み型アプリケーションの開発環境



【書類名】 要約書

【要約】

【課題】 OSシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のOS管理情報を参照できるようにしたデバッグ支援システムを提供する。

【解決手段】 共有ファイルFには、OSシミュレータ20の要請に応じてOS管理情報が書き込まれ、このOS管理情報は、OSデバッガ40の要請に応じて読み出される。すなわち、アプリケーション10側に備えたOS管理情報書き込み手段12が共有ファイルFにOS管理情報を書き込み、OSデバッガ40側に備えたOS管理情報読み出し手段43が共有ファイルFからOS管理情報を読み出すようになっている。このようにすれば、アプリケーション10が停止した場合でも、この時点のOS管理情報を共有ファイルFから参照できる。

【選択図】 図1

特 2 0 0 0 - 0 9 4 0 6 4

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 8 2 1]

1. 変更年月日 1 9 9 0 年 8 月 2 8 日

[変更理由] 新規登録

住 所 大阪府門真市大字門真 1 0 0 6 番地

氏 名 松下電器産業株式会社